Assignment 5-1 Prime Numbers

A *prime number* is a positive integer that has only two factors, that is, 1 and the number itself. For example, 2, 3, 5, 7 are prime numbers. A *composite number* is a positive integer that can be formed by multiplying two smaller positive integers other than 1. Equivalently, it is a positive integer that has at least one factor other than 1 and itself.

Input

The input consists of t ($30 \le t \le 40$) test cases. The first line of the input contains only positive integer t. Then t test cases follow. Each test case consists of exactly one line with an integer n ($2 \le n \le 10^7$).

Output

For each such integer n, you are to output a single line containing the word "prime" or "composite" depending on whether the integer n is a prime number.

Sample Input

```
7
2
3
4
5
6
7
8
```

Sample Output

```
prime
prime
composite
prime
composite
prime
composite
```

Requirements

You are required to write a recursive function `bool hasFactor( int n, int end )` to complete the following program which solves this problem. This function returns true if and only if `n` has a factor in { 2, 3, ..., `end` }.

```
#include <iostream>
#include <cmath>
using namespace std;

// returns true if and only if n has a factor in { 2, 3, ..., end }
bool hasFactor( int n, int end );

int main()
{
   int numCases;
   cin >> numCases;
   for( int i = 1; i <= numCases; i++ )
   {
      int n;
      cin >> n;

      if( n == 2 || n == 3 )
         cout << "prime" << endl;
      else if( hasFactor( n, static_cast< int >( sqrt( n ) ) ) )
         cout << "composite" << endl;
      else
         cout << "prime" << endl;
   }
}

// returns true if and only if n has a factor in { 2, 3, ..., end }
bool hasFactor( int n, int end )
{


}
```